

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許出願公告番号

特公平6-19731

(24)(44)公告日 平成6年(1994)3月16日

(51)Int.Cl. ⁵	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 11/28 9/45	3 4 0 A	9290-5B 9292-5B	G 0 6 F 9/ 44	3 2 2 C

発明の数1(全 8 頁)

(21)出願番号 特願昭59-250174
(22)出願日 昭和59年(1984)11月27日
(65)公開番号 特開昭61-127044
(43)公開日 昭和61年(1986)6月14日
審査番号 平4-10826

(71)出願人 999999999
富士通株式会社
神奈川県川崎市中原区上小田中1015番地
(72)発明者 柳沢 実
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内
(74)代理人 弁理士 山谷 皓榮

審判の合議体
審判長 松尾 浩太郎
審判官 大橋 隆夫
審判官 吉岡 浩

(56)参考文献 特開 昭58-175061(JP, A)
「新版 情報処理ハンドブック」情報処
理学会編 昭和55年3月30日 オーム社発
行 P. 252-260

(54)【発明の名称】 プログラムの単体テスト方式

【特許請求の範囲】

【請求項1】複数のモジュールが階層的に構成されるプログラムのモジュール単体を、プロセッサが実行してテストを行うプログラムの単体テスト方式において、該被テストモジュールを解析して、上位及び下位モジュールの存在を検出して、上位又は下位のモジュールの代替機能を行うテスト支援ツールを設け、該テスト支援ツールは、該上位モジュールの存在を検出した場合には、該被テストモジュールに渡すパラメータの領域をメモリに確保して、テストデータをセットした後、該被テストモジュールを起動せしめ、該下位モジュールの存在を検出した場合には、該メモリにパラメータ領域を得て、該被テストモジュールのテスト実行中に、該下位モジュールが呼び出されたことに応じて、該被テストモジュールに渡すテストデータを該パ

ラメータ領域にセットして、該被テストモジュールに渡すことを特徴とするプログラムの単体テスト方式。

【発明の詳細な説明】

〔産業上の利用分野〕

本発明は、複数のモジュールで構成されるプログラムの各モジュールプログラムを単体でテストするプログラムの単体テスト方式に関し、特に関連する他のモジュールが作成されていなくてもテストできるプログラムの単体テスト方式に関する。

一般に大きなシステムにおいては、プログラムが1本のモジュール(コンパイル単位)で構成されているものは少なく複数のモジュールが集まって一つの処理を行なうように階層的に構成されている。

例えば、第5図に示すようにモジュールAのサブルーチンとしてモジュールBが、モジュールBのサブルーチン

としてモジュールC、Dが構成されて1つのプログラムを構成するようにしている。このような階層的なプログラムを開発するには、テストの効率を考慮してモジュールの作成順序等が決定され、一般に第6図で示す如くシステム設計された後、各モジュールA、B、C、Dが作成される。

この開発の手順としては、上位のモジュールAから作成するトップダウン方式と、下位のモジュールC又はDから作成するボトムアップ方式とがある。

〔従来の技術〕

このような各モジュールが階層的に構成されたプログラムをテストするには、全てのモジュールが完成してからプログラムをテストすることが望ましいが、各モジュールが順次作成されていくような場合には、各モジュールの作成毎にそのモジュールを単体でテストするほうが開発手順上効率がよい。

しかしながら、モジュールを単体でテストするにしても、階層的なプログラムの構成では、関連する他のモジュールが存在しないとテストできない。例えばモジュールAをテストするときは、モジュールBが必要であり、モジュールBをテストするときは、モジュールA、C、Dが必要となる。このため、従来は、テストするモジュールに関連する仮モジュールをテスト担当者が作成し、係るモジュールの単体テストを行っていた。

例えば、モジュールBをテストするため上位のモジュールAの仮モジュール（ドライバという）及び下位のモジュールC、Dの仮モジュール（スタブという）を作成していた。

〔発明が解決しようとする問題点〕

このようなモジュールのテストで必要となるドライバやスタブは仮モジュールなので小さく簡単に作成できるが、通常のプログラム作成と同様にコーディングやこれのテストの工数を要し、単体テストのためのドライバやスタブの開発をモジュール個々に要するという問題があった。

これを解決するため、被テストモジュールの下位モジュール代替機能を行うスタブ機能を設けたテスト方法が提案されている（例えば、特開昭58-175061号公報等）。

このようにしても、次の問題があった。

- ①上位モジュールの代替機能がないため、トップダウンプログラミングしか適用できず、ボトムアッププログラミングや両者の混在における単体テストができない。
- ②下位モジュールの代替時に、使用領域等を設定する必要がある、データ設定に手間がかかる。

〔問題点を解決するための手段〕

従って、本発明は、被テストモジュールを解析して、上位、下位モジュールの存在を検出して、上位、下位の代替機能を自動設定することができるプログラムの単体テスト方式を提供することを目的とする。」

このため、本発明は、複数のモジュールが階層的に構成されるプログラムのモジュール単体を、プロセッサが実行してテストを行うプログラムの単体テスト方式において、該被テストモジュールを解析して、上位及び下位モジュールの存在を検出して、上位又は下位のモジュールの代替機能を行うテスト支援ツールを設け、該テスト支援ツールは、該上位モジュールの存在を検出した場合には、該被テストモジュールに渡すパラメータの領域をメモリに確保して、テストデータをセットした後、該被テストモジュールを起動せしめ、該下位モジュールの存在を検出した場合には、該メモリにパラメータ領域を得て、該被テストモジュールのテスト実行中に、該下位モジュールが呼び出されたことに応じて、該被テストモジュールに渡すテストデータを該パラメータ領域にセットして、該被テストモジュールに渡すことを特徴とする。

〔作用〕

本発明では、テスト支援ツール（プログラム）が設けられるが、この支援ツールがドライバ又はスタブ機能を自動的に実行するようにしている。即ち、支援ツールは被テストモジュールを解析し、これに含まれる上位又は下位モジュールのインターフェイス（接続）情報を抽出し、上位又は下位モジュールの存在を検出し、上位モジュールが存在する時は被テストモジュールの起動に先立ち、インターフェイス情報を基にパラメータ領域をメモリに確保し、更にこの領域に外部から与えられたテストデータをセットして被テストモジュールのテスト起動を行ない、又下位モジュールが存在するときは、該被テストモジュールのテスト実行中に下位モジュールを呼出すことに応じてインターフェイス情報から外部から与えられたテストデータをメモリにセットして被テストモジュールに渡すようにして被テストモジュールのテストを続行させるようにしている。

〔実施例〕

以下、本発明を実施例により詳細に説明する。

第1図は本発明の一実施例ブロック図であり、図中、1はコンピュータ（プロセッサ）であり、プログラムを実行して所望の処理を行なうもの、10はそのOS（オペレーティングシステム）部であり、コンピュータ1全体の制御を行なうもの、11はモジュール解析部であり、後述する如く論理設計言語で記述された論理設計情報を解析してCOBOLのソースプログラムを作成し、且つインターフェイス（デバッグ）情報を抽出するもの、12はCOBOLコンパイラであり、ソースプログラムをコンピュータの実行できる形式のロードモジュールに変換するもの、13はテスト支援ツール部であり、後述する如くインターフェイス情報からドライバ／スタブ機能を行なうものである。尚、OS部10、モジュール解析部11、COBOLコンパイラ12及びテスト支援ツール部13はコンピュータ1の実行する機能をブロック化したものである。2はファイルメモリであり、各種ファ

イルを格納しておくもの、20は論理設計情報ファイルであり、プログラマーが作成した後述する論理設計情報を格納しておくもの、21はCOBOLソースファイルであり、モジュール解析部11によって作成されたCOBOLソースプログラムを格納しておくもの、22はデバッグ情報ファイルであり、モジュール解析部11で抽出されたインターフェイス情報を格納しておくもの、23はロードモジュールファイルであり、COBOLコンパイラ12が作成したロードモジュールを格納しておくもの、24はテストデータファイルであり、オペレータが入力したテストデータを格納しておくもの、3はコンソールであり、ディスプレイとキーボードから成り、オペレータが操作して論理設計情報やテストデータを入力するものである。

次に、第1図実施例構成の動作について第2図の処理フロー図、第3図の論理設計情報説明図及び第4図のテスト支援ツールの動作説明図を用いて説明する。

以下、周知の高級言語である論理設計言語によりプログラミングされたものについてその動作を説明する。

①論理設計言語によるプログラミングは、第3図に示す如く各モジュールA、B、C、Dで共通なフォーマット定義書(第3図(A))、各モジュールのインターフェイス定義書(第3図(B))、各モジュールのモジュール定義書(第3図(C))等を作成することにより行なう。この共通フォーマット定義書は各モジュールで共通な項目(データ)の形式、桁数等の定義を行なうものであり、インターフェイス定義書は各モジュールの入出力項目、モジュール名等を定義するためのものであり、基本設計段階で決定され、コンソール3よりコンピュータ1を介しファイルメモリ2の論理設計情報ファイルに格納されている。

一方、モジュール定義書は各モジュール毎に作成され、当該モジュールの処理や論理、その条件が手順に従って羅列されたものであり、プログラマーは各モジュールに必要な処理等を順次記述してプログラミングし、これが当該モジュールの処理内容を示す。オペレータはこのモジュール定義書を完成する毎にコンソール3からコンピュータ1を介し論理設計情報ファイル20に格納せしめる。

例えば、モジュールBのモジュール定義書が完成し、論理設計情報ファイル20に格納され、他のモジュールA、C、Dは完成していないとする。

②オペレータはコンソールよりコンピュータ1にOS部10のジョブ制御機能を働かせ、モジュールBのCOBOLソースプログラムを作成するように指令する。これによってコンパイラの一つであるモジュール解析部11は論理設計情報ファイル20の論理設計情報を読み出し、COBOLで記述されたソースプログラムを作成し、COBOLソースファイル21に格納する。このCOBOLで記述されたソースプログラムは、例えば第4図の被

テストモジュールとして示すものであり、論理設計言語を用いずプログラマーがCOBOLでプログラミングしたものに相当する。

③これとともにモジュール解析部11は論理設計情報からインターフェイス情報を抽出する。例えば、第3図の例では、モジュールBはモジュールAに呼び出され、パラメータの数はG1、G2の2つでその領域はA1、A2であることがモジュールBのインターフェイス定義書より抽出され、そのパラメータG1、G2の長さは共通フォーマット定義書より抽出される。又、モジュールBのモジュール定義書よりモジュールC(日数=%日数計算)、モジュールDを呼び出すことが抽出され、その使用領域はモジュールC、Dのインターフェイス定義書より抽出される。

これをCOBOLソースプログラムの記述から見ると、第4図の被テストモジュールの記述の如く、モジュールAのA1、A2を用い、モジュールCのB1、モジュールDのB2を用いることになる。

モジュール解析部11はモジュールBが呼び出されるときにインターフェイス(モジュール名、パラメータの数、パラメータの領域名、その長さ等)及びモジュールBが呼び出す時のインターフェイスを論理設計情報から抽出し、デバッグ情報ファイル22に格納する。

このようにしてモジュールBに対するCOBOLソースプログラムが作成され、且つモジュール間のインターフェイス情報が抽出される。

④次に、モジュールBの単体テストを行なう。

このため、オペレータはコンソール3よりコンピュータ1を介しモジュールBのテストのためのテストデータをテストデータファイル24に格納する。そして、オペレータはテスト実行指令をコンピュータ1に入力する。コンピュータ1では、OS部10がCOBOLコンパイラ12を動作せしめ、COBOLソースファイル21のモジュールBのCOBOLソースプログラムをコンピュータ1の実行できるロードモジュールに翻訳してロードモジュールファイル23に格納する。次にOS部10はテスト支援ツール部13を起動し、デバッグ情報ファイル22のインターフェイス情報を解析せしめる。

⑤テスト支援ツール部13は、第2図の処理フロー図の如く前述のインターフェイス情報を解析し、上位モジュールが存在するかを検出する。上位モジュールが存在するとドライバ機能、即ち上位モジュール代替機能を実行する。先づ、被テストモジュールBに渡すパラメータの受け渡し領域をコンピュータ1の図示しない内部メモリに確保する。前述の例では、第4図に示す如く内部メモリにA1及びA2の入出力領域を確保し、そのアドレスをセットしておく。

次に、コンソール3よりオペレータにテストデータの指示を表示し、オペレータはテストデータファイル24のどのテストデータをセットするかをコンソール3より指

示する。これによって内部メモリのA1の領域にテストデータファイル24の必要なテストデータ（上位モジュールからの入力データ）がセットされることになる。

これによってモジュールBはパラメータA1を用いて処理を実行できる。尚、A2はモジュールBの実行結果の出力のためのものである。

⑥次に、テスト支援ツール部13はOS部10に対し被テストモジュールの起動を指示する。

OS部10はロードモジュールファイル23のモジュールBのロードモジュールを実行し、その中でモジュールAのパラメータA1を用いて行なう。

⑦このようにしてモジュールBのテストを実行していくうちに、モジュールC又はDの“call”（呼び出し）命令があり、OS部10がロードモジュールファイル23を探索した結果モジュールC、Dのロードモジュールが見付からなかったとすると、エラーをテスト支援ツール部13に通知する。

⑧これによってテスト支援ツール部13は、被テストモジュールが下位モジュールを呼ぶために用意した内部メモリのパラメータ領域（B1やB2）のアドレスを得、コンソール3よりオペレータにテストデータの指示を表示する。オペレータはテストデータファイル24のどのテストデータをセットするかをコンソール3より指示する。これによって内部メモリのパラメータ領域にテストデータファイル24の必要なテストデータ（下位モジュールの出力データ）がセットされることになる。

従って被テストモジュールBはこのテストデータを下位モジュールの出力データとして受け取り、更にテストを続行していく。

このようにして、テスト支援ツール部13が、インターフェイス情報から上位又は下位モジュールの代替機能を自動的に実行し、被テストモジュールのテストを行なわしめる。

上述の実施例では、論理設計言語でプログラムした例について説明したが、COBOLによってプログラムしてソースプログラムを作成してもよく、この場合COBOLソースプログラムからインターフェイス情報が抽出さ

れる。又、テストデータをパラメータ領域にセットするのにコンソール3よりオペレータが指示しているが、これを自動的に行なってもよい。更にソースプログラムの言語もCOBOLに限らず、他の言語を用いてもよい。以上本発明を一実施例により説明したが、本発明は本発明の主旨に従い種々の変形が可能であり、本発明からこれらを排除するものではない。

〔発明の効果〕

以上説明した様に、本発明によれば、次の効果を奏する。

①上位及び下位モジュールの存在を検出して、上位又は下位のモジュールの代替機能を行うテスト支援ツールを設けたので、トップダウンプログラミング、ボトムアッププログラミングのいずれでも、単体テストが可能となる。

②被テストモジュールを自動解析して、上位及び下位モジュールの存在を検出して代替機能を行うので、自動的に上位、下位の代替動作が実行できる。

③上位モジュールの存在を検出した場合には、被テストモジュールに渡すパラメータの領域をメモリに確保して、テストデータをセットした後、被テストモジュールを起動せしめるので、上位モジュールの代替機能を自動的に実行できる。

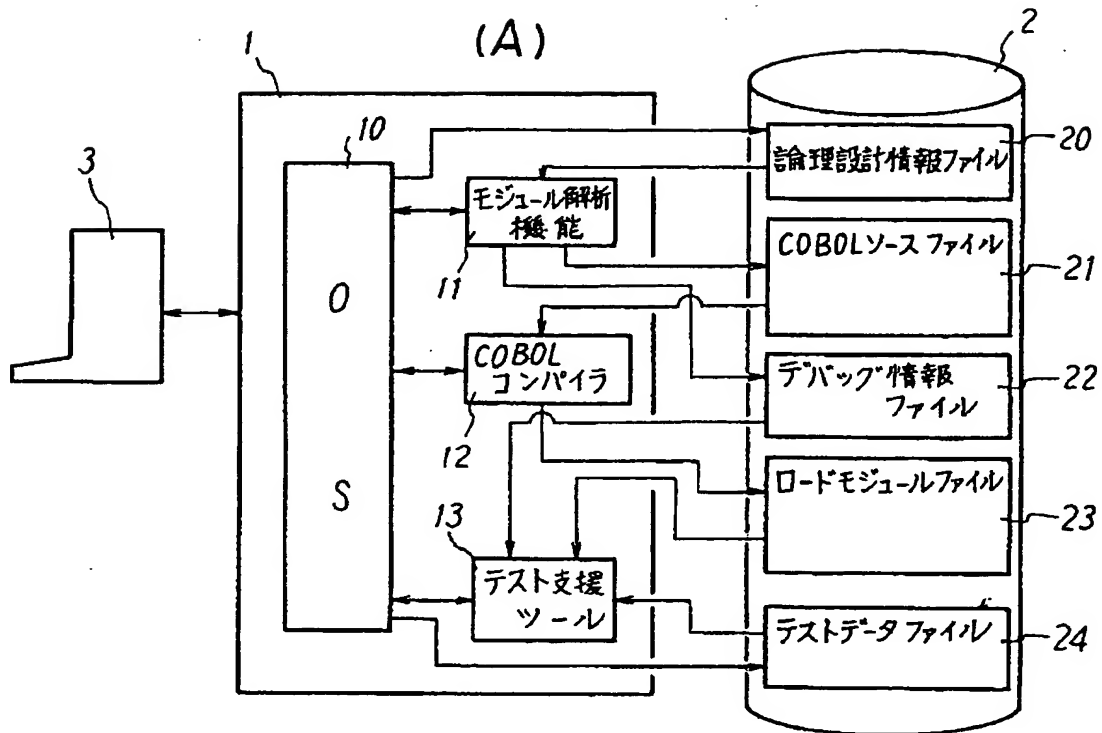
④下位モジュールの存在を検出した場合には、メモリにパラメータ領域を得るので、データ設定が容易となり、単体テストを高速に実行できる。

【図面の簡単な説明】

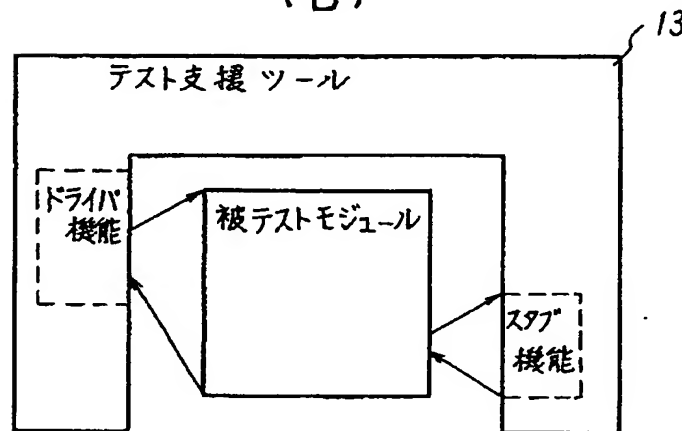
第1図は本発明の一実施例ブロック図、第2図は第1図実施例構成のテスト支援ツールの処理フロー図、第3図は第1図実施例構成に用いられる論理設計言語の説明図、第4図は第1図実施例構成のテスト支援ツールの動作説明図、第5図は階層的モジュール構成プログラムの説明図、第6図は第5図構成のプログラムの開発手順説明図である。

図中、1……コンピュータ（プロセッサ）、2……ファイルメモリ、3……コンソール、13……テスト支援ツール。

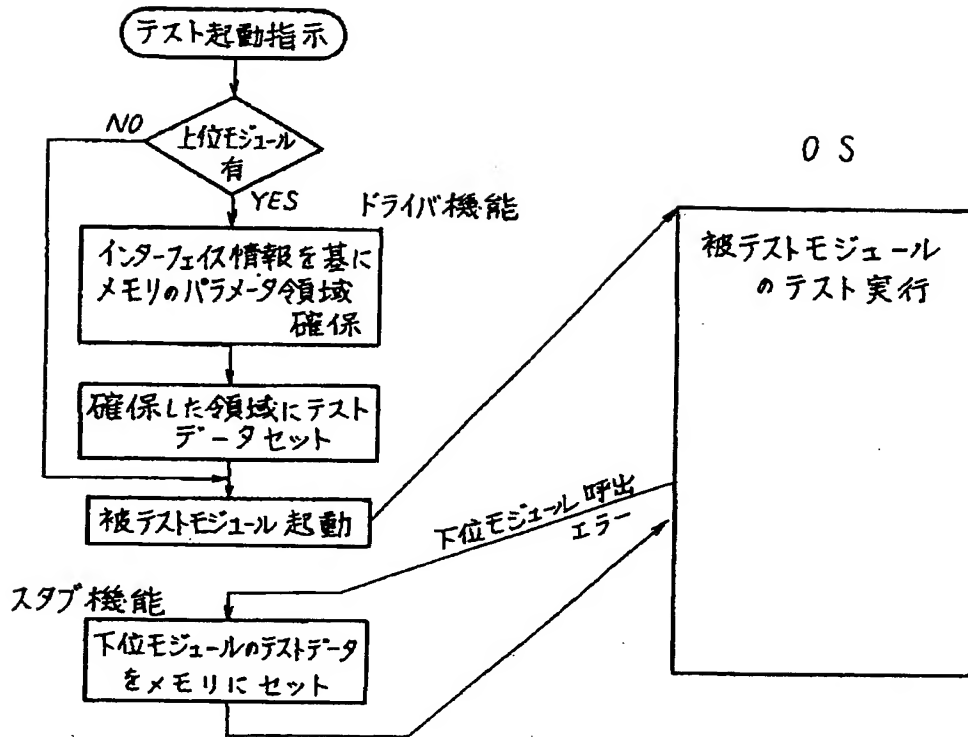
【第1図】



(B)



【第2図】



【第3図】

(A)

共通フォーマット定数

項番	区分	項目	形式	桁
10		口座番号	9(4)	
20		科目コード	9(4)	
30		氏名	N(6)	

(B)

インターフェイス定義 (モジュールB)

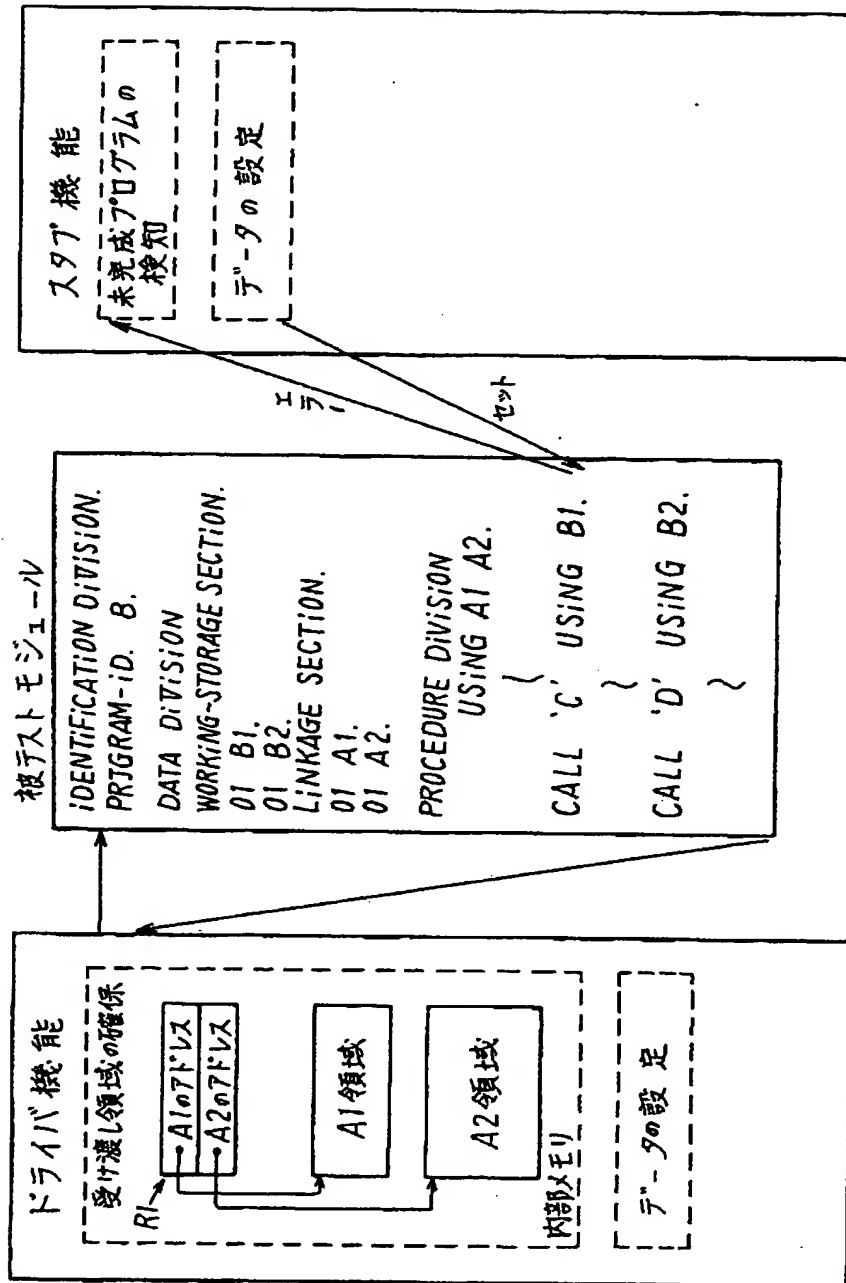
項番	区分	インターフェース	入出力項目
1	S	マシ管理 B	GRP: G1 = A1 G2 = A2

(C)

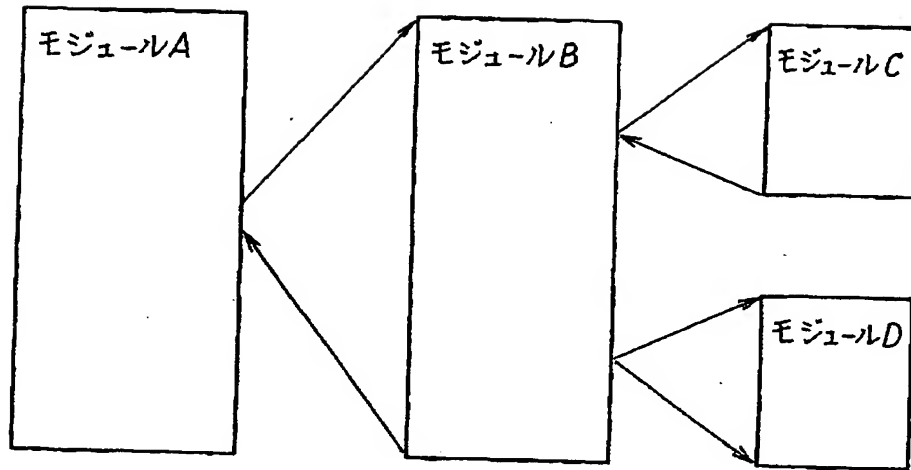
モジュール定義 (モジュールB)

項番	区分	処理	論理	条件
10				
20				
30		日数 = %日数計算 C		

【第4図】



【第5図】



【第6図】

